Imitation Learning for UAS Navigation in Cluttered Environments

Caleb Harris*, Youngjun Choi† and Dimitri N. Mavris‡
Aerospace Systems Design Laboratory, School of Aerospace Engineering,
Georgia Institute of Technology, Atlanta, GA, 30332-0150, USA

Autonomous navigation is a critical component for the use of unmanned aerial systems (UAS) in complex tasks such as package delivery and disaster response. In recent years, these systems have seen increased usage in harsh tasks such as search and rescue and disaster relief, however there remains challenges for efficient and safe operation in a fully autonomous mission. This work seeks to provide a data-driven, vision-based method to navigating and searching through a clustered environment, which is high-speed, low-cost and vehicle-agnostic. This is done by first assuming obstacle avoidance as a two-dimensional navigation task that can be solved by knowing the relative location of the goal and the 2D image of the obstacle in the camera frame. Imitation learning is used to train a deep neural network from an expert planning policy, while a model predictive controller tracks the target. All the processing is capable onboard the vehicle, with the assumption that the general target direction is forward of the camera-frame, and that the global state estimation error is low. The framework and trained model are tested in simulation, with a quadcopter conducting search scenarios in different environments. The resulting framework is quicker to avoid obstacles and can be applied on small, low-cost systems with a single monocular camera.

I. Nomenclature

UAS	=	unmanned aerial system
SAR	=	search and rescue
Q	=	state-error cost matrix
R	=	input cost matrix
MPC	=	model predictive control
DNN	=	deep neural network
A^*	=	A-star search algorithm
π	=	learned policy
β	=	DAgger policy probability

II. Introduction

The use of unmanned aerial systems (UAS) has grown rapidly in recent years with advances in onboard sensors, control algorithms, and communication architectures. UAS are used for tasks often referred to as "dull, dirty, dangerous". Examples of common tasks include package delivery, construction surveillance and monitoring, search and rescue, and disaster response [1]. These tasks are suited for UAS because of their agile flight and small footprint. Furthermore, these systems improve safety and consistency, as a human can be removed from some, if not all, of the required tasks. The successful operations of UAS paralleled by technological advancement and increasing public acceptance has pioneered an active field.

The United Nation's Office for the Coordination of Humanitarian Efforts (OCHA) produced a report in 2014 outlining the previous applications and future potential of this industry citing examples of UAS use during disasters in

*Graduate Research Associate, Aerospace Systems Design Laboratory, School of Aerospace Engineering, 270 Ferst Drive, Atlanta, GA, 30332-015

†Research Engineer II, Aerospace Systems Design Laboratory, School of Aerospace Engineering, 270 Ferst Drive, Atlanta, GA, 30332-0150

‡S.P. Langley Distinguished Regents Professor, Boeing Regents Professor of Advanced Aerospace Systems Analysis, School of Aerospace Engineering, Director Aerospace Systems Design Laboratory, 270 Ferst Drive, Atlanta, GA, 30332-0150, AIAA Associate Fellow

the Philippines, Haiti, and the Democratic Republic of Congo [2]. The report outlines the requirements and capabilities of the potential use of UAS for damage data collection, medicine package delivery, and peacekeeping. A few recent incidents have shown the successful use of UAS such as the collapse of the Hard Rock Hotel in New Orleans, where first responders used a drone to survey the stability of the building before entering to search for survivors *. Other examples include in Houston, Texas where swarms of remotely-piloted UAS were used, primarily for surveying critical infrastructure† and in Sherburne County, Minnesota where a young kid was found late at night using a thermal camera mounted on a remotely-piloted quadcopter‡. These scenarios all required a human pilot to control and make the decision for the UAS, where now there is heightened interest in how to remove the human to be able to more efficiently and rapidly send out these systems autonomously. Though, it is clear that autonomous systems can play a critical role as life-saving technology in SAR.

The use of autonomous systems in these scenarios has been limited due to the difficulty in navigating and making decisions in a dynamic and diverse environment. Problems include limited GPS, disruptive sensor noise and unknown obstacles, examples of which are shown in 1. In order to navigate through these environments, UAS must be capable of a high-level of actionable intelligence. However, with cost limits, size constraints, and incomplete knowledge a priori, the systems must be able to make decisions with limited information and processing. The task of navigation and exploration involves feedback from the environment in the form of observations from sensors. In the case of exploring a maze, a general robot may know the basic ideas of where the walls are, and thus be capable to search for the target position. However, if a new obstacle appears that was either unknown or moved, then the robot may be unable to avoid the obstacle along the path. This can decrease the success of the mission, or cause it to fail completely. As has been known for years, the capability of obstacle detection and avoidance is crucial to the safety and success of missions involving navigation.

A major limitation of autonomous system obstacle avoidance is the amount of computational power available. This includes onboard computational power, which must fit and be powered on the system during operation. Another limitation is the time constraints in the scenarios mentioned earlier. One, the latency effects from communication and processing requirements can be detrimental to the control and navigation algorithms. Two, some tasks have hard time constraints, which must be met for success of the mission. Another limitation is the amount of data from the environment that is available from the onboard sensors. The sensors used by UAS are often small and low-cost, thus the information provided is limited. Additionally, the scenarios in which they are applied often have detrimental effects on the sensors. Lastly, the capability is limited by the diverse set of potential scenarios, which the autonomous system must be able to handle.

Technical limitations have delayed the implementation of UAS into more complex environments. Therefore, there is a need for an improved method of obstacle avoidance and navigation for search operations. This work demonstrates a methodology for navigation and search where sensor information is limited. The method displays improvements in speed, cost, and performance. This is done by reviewing the requirements of search and rescue and disaster response missions by UAS and then testing with currently available systems and methods to determine the inadequacies. An investigation into related works shows the potential of reinforcement learning methods and a methodology is outlined in how these methods are used. A results section describes the case study and the capabilities of the method.

III. Background

The problem can be treated as a path planning or optimal control problem, wherein the goal is to find the feasible path with the least cost. This problem can involve obstructions, but is usually then referred to as obstacle avoidance. In further breaking down the problem, obstacle avoidance can be divided into a perception phase, a planning phase, and a control phase. This work focuses on a combined perception and planning phase for the obstacle avoidance procedure. During the avoidance procedure, the problem usually becomes more complicated with a lot of noise in the perception phase and additional constraints and the lost of convexity for planning. A review of obstacle avoidance and modern methods of solution are detailed in the following sections.

A. Methods of Obstacle Avoidance

Obstacle avoidance can be divided into four areas: stochastic, road map, potential field, and geometric-optimization methods. Each has pros and cons and the decision to use any one is highly dependent on the situation. There are endless

*https://www.nola.com

†https://www.wired.com

‡https://www.washingtonpost.com

Fig. 1 Examples for the Need of Obstacle Avoidance in UAS for SAR

examples of each being used in research and in industry. Stochastic methods involve the use of random path building algorithms, which rely on the likelihood that a good solution exists. An example would be the Rapidly exploring Random Tree (RRT) algorithm. This method benefits from the fact that it is not limited by complex, nonconvex, high-dimensional space, and it has a fast computation time, dependent however on limiting the number of paths to search. However, the solution could be highly suboptimal and is not guaranteed to converge. Potential field methods consist of defining the system and any obstacles or targets to be represented as particles in a force field. Therefore, the system can be pushed or pulled to different areas on the map depending on the sensor information. This method is easy to implement and requires minimal sensor information, however the parameters must be tuned for each situation and vehicle, and a collision-free path is not guaranteed. Road map methods consider the environment as a grid map or a graph, and uses the information on the system, obstacles, and goal to define costs to each cell or node. There are many algorithms for this, such as A*, which defines the value of each cell or node based on a heuristic, then biases the search direction based on the value of the heuristic in each cell. This process guarantees a collision free path if one exists, but the optimality depends on the heuristic used. Geometric methods are often used with optimization methods, such as in the case of forming an optimal trajectory with constraints defined for conflict regions. In this method, optimal solutions are based on user preference defined by the cost function, and if the geometric representation is accurate then a collision-free path is guaranteed. However, this is complex to implement and vehicle dependent.

The methodology behind an obstacle avoidance or navigation algorithm involves the sensors, the architecture, and the mathematics behind the solution. The following section details previous works and their strengths and weaknesses. Choi developed a framework for two-layer collision avoidance for a fixed-wing UAS and conducted tests in simulation [3] [4]. A LIDAR sensor and the DBSCAN algorithm provided a point cloud representation of the obstacles, and then a geometry optimization technique was used in a multi-phase approach to avoid multiple obstacles in an environment. The method was applied onboard the vehicle using Model Predictive Control. Successful demonstrations proved the capability of this technique, however the computational requirements limited the use to simulation. Barry used a stereo camera with limited processing power to track trajectories and avoid obstacles at high speed onboard a fixed-wing UAS [5]. In this case, the trajectory around the object was not of interest, but the focus was on flying in a direction away from the expected relative location of the obstacle. Hrabar in [6] used a dynamic version of A*, called D*-lite, to apply onboard path planning using on a rotorcraft with a stereo camera. This work showed the potential for road map methods applied to onboard aerial systems, though the system was only tested on a suspended system and was never applied in real flight.

B. Data-driven Techniques for Autonomous Systems

In the early 2000s, reinforcement learning showed its potential for autonomous systems and robotics, for example two papers by Michels [7] and Abbeel [8]. Michels used a monocular camera on a remote control car to avoid obstacles at high speed by learning the object depths from the camera image. Abbeel used pilot training data and imitation learning to have an aerobatic helicopter perform flips and inverted flight. In the last few years successful experiments have been completed in this area. Pan et al implemented an end-to-end policy, which directly maps raw sensor data to control inputs, using a deep neural network and image data [9]. The neural network was applied on a rally-car for racing

around a dirt track. Pan et al used the DAgger method of imitation learning and a model predictive controller using the differential dynamic programming (DDP) solver technique. Pfeiffer et al utilized pre-training of the policy using imitation learning, then continues the learning approach usign model-free methods to generalize the policy to unseen and real-world environments [10]. Zhang et al used model predictive control supplied with global state information as the expert to train a policy in simulation [11]. Kaufman et al applied a convolutional neural network on a high speed racing drone to learn obstacle locations even in noisy situations [12]. Similar research as been conducted for crowded environments, in particular forested areas, such as in [13] and [14]. The general approach in these two works is the use of a normalized output in the range of -1 and 1 which directs the robots steering angle, which in the case of a quadrotor would be translated to motor speeds by a low-level controller, most likely a PID controller.

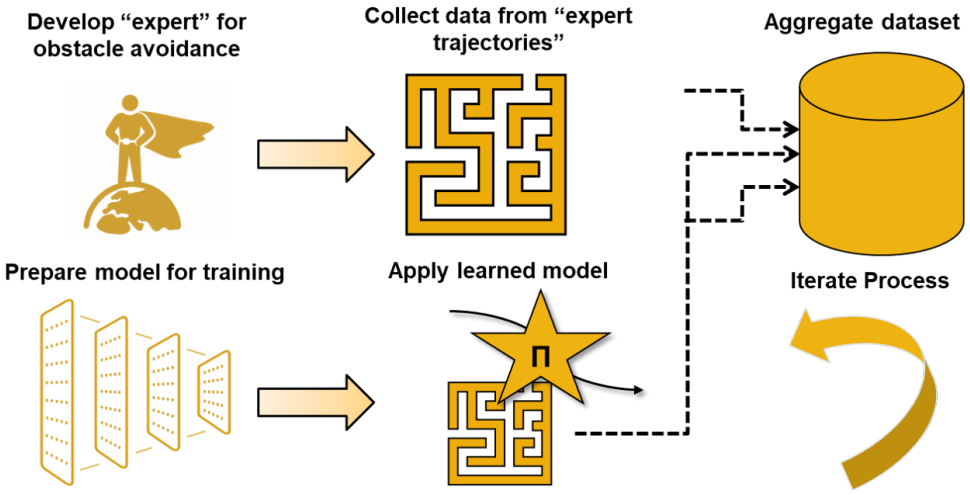

Fig. 2 Overview of Methodology

IV. Methodology

The literature review outlined four key outcomes of the various learning and obstacle avoidance methods that provide the potential for simple implementation techniques, which can be applied to generalized scenarios in complex environments. One, deep neural networks have wide-latitude to learn complex behaviors. Two, imitation learning provides a method to quickly train policies to perform well in specific scenarios. Three, model predictive control using optimal control methods provides a great expert controller for training. Four, vision-based techniques have shown successful applications in many robotic systems.

There is clearly a critical need of onboard obstacle avoidance onboard for UAS search and rescue applications in complex environments. This work seeks to develop a framework and baseline model for a small aerial system avoiding a set of obstacles in a generalized environment. The model is developed using imitation learning and is compared to the expert system in terms of performance, complexity, and sensitivity.

The use of reinforcement learning has led to many exciting systems, but the ability to train policies without worrying about safety and system limitations is not a privilege of physical robot implementations. Therefore the reinforcement learning structures applied must be knowledgeable of the requirements in implementation and limitations of the system. In addition, the ability to scale the policy to other systems is important, as UAS can come in many sizes, shapes, and with various sensors. In the case of UAS obstacle avoidance, the hierarchy of decision-making is dependent on where data-driven techniques are applied. For instance, a UAS flying through an environment may have a series of algorithms running onboard: tasking, path-planning, high-level control, low-level control. Decisions are being made within each of these loops, such as the next waypoint location to move towards or the desired motor speeds. The level at which a learned-policy is used should be dependent on the requirements and limitations of the decision. For instance, if a learned-policy is choosing the motor speeds for the UAS, then it must be able to update the decisions at a faster rate than the dynamics of the UAS, and it must also be aware of the speed and response limits of the motors. Another important aspect of a learned-policy, is understanding in what scenarios it applies. For instance, whether or not the

policy is effective in scenarios requiring a different route, or with obstacles that were not in the training data. The primary question here is how does the learned policy perform in comparison to the effect that noise has on traditional obstacle avoidance method.

The methods of implementation are shown in Figure 2 and outlined in the following sections. A data-driven technique is used to provide a simple and fast, yet effective method of obstacle avoidance. The DAgger imitation learning method is used to combine the safety and sample-efficiency of supervised imitation learning and the ability to generalize from reinforcement learning. The data-driven process is chosen to provide the path planning for obstacle avoidance to improve overall processing speed and be vehicle-agnostic, while the trajectory planning and controls are handled by a nonlinear model predictive controller and a PID controller. The A^* algorithm is selected as the expert policy to train a deep neural network to output the best relative waypoint between the current location and the next target waypoint.

A. Dynamics Model and Controllers

The UAS model used is a quadcopter as seen in Figure 3 and is a slightly modified version of the work from [15]. The world reference frame W and the body fixed frame B define the systems movement. The rotation between the two is defined by the rotation matrix R_{WB}. The body forces acting on the system can be described proportionally to the square of the rotation speed of the propellers, ω^2, by the constant k_T.

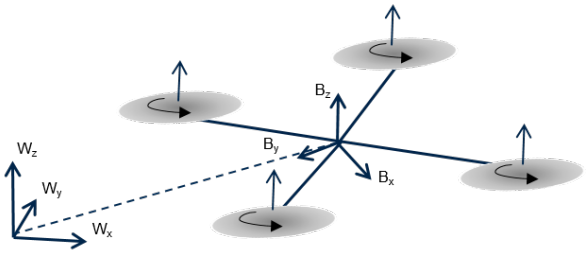

Fig. 3 Quadcopter model for flight simulation

The dynamics model must account for the inner loop dynamics for trajectory tracking. Therefore, a model of the low-level attitude controller is constructed with the desired angular rate of the system as the input, and the gain, k, and time constant, τ, as the tuning parameters. The difference from previous work is the tracking parameter being angular acceleration, rather than angular velocity, and the reference parameter being angular velocity, rather than the angle in radians. This implementation was based on a tradeoff between responsiveness and safety within the Model Predictive Controller. Experiments show that bounded $\dot{\theta}$ and $\dot{\phi}$ values as tracking targets keep the system in stable flight conditions, while also improving the reaction time during quick maneuvers.

$$\ddot{\phi} = \frac{1}{\tau_\phi}(k_\phi \dot{\phi}_{cmd} - \dot{\phi}_{ref}) \tag{1}$$

$$\ddot{\theta} = \frac{1}{\tau_\theta}(k_\theta \dot{\theta}_{cmd} - \dot{\theta}_{ref}) \tag{2}$$

Model-based control and optimal control theory provide methods of control which are more efficient and accurate. The optimal control problem is defined by a running cost, a positive-definite state-error cost matrix Q, an input action cost, a positive-semi-definite input cost matrix R, and a terminal cost, a positive-definite terminal state-error cost matrix Q_f. The objective function is shown in Equation 3, with the goal to minimize the cost by modifying the control input, u.

$$\min_u J(u,x) = (x_{t_f} - x^*_{t_f})^T Q_f (x_{t_f} - x^*_{t_f}) + \int_{t_0}^{t_f} (x_\tau - x^*)^T Q (x_\tau - x^*) + (u_\tau - u^*)^T R (u_\tau - u^*) \tag{3}$$

A Model Predictive Controller (MPC) is used for determining the optimal control trajectory for the UAS. The MPC receives state feedback from the onboard sensors and then solves the optimal control problem with the desired setpoint, cost function, dynamics model, and constraints. For this work, the ACADO toolkit[§] is used to convert a C++ template

[§]http://acado.github.io/

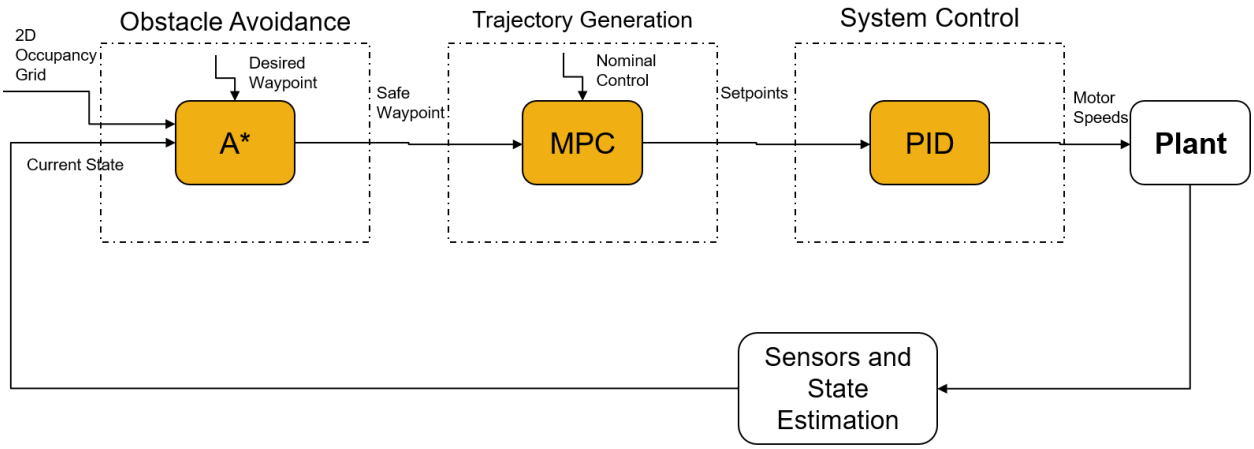

Fig. 4 System architecture for the expert controller

file to generate optimized C-code for the specific nonlinear optimal control problem. This work featured a sampling time of 0.1 seconds, horizon of 20, and a multiple shooting solver with a gauss-newton dynamics approximation.

Considering the MPC is acting as a trajectory generator of the angular acceleration and throttle commands of the system, a low-level controller is required to follow the commands at a high rate. A PID controller is used because of the quick computational time and the ability to tune for specific flight manuevers, like landing and avoidance.

B. Expert Path Planner

The collision free path is found by using the road-map method, A*. The onboard stereo camera produces a point cloud using stereo feature-matching. The point cloud is downsampled into Octrees, and then a cross-section at the system altitude provides the 2D occupancy grid. The A* algorithm then searches the grid, starting from the current position and ending at the goal location. At each step forward the algorithm looks for a cell that is the lowest sum of the distance from the start point and distance to the goal point. This works as a trade-off of Breadth-First search and Greedy Best-First search.

The full architecture for the expert can be seen in Figure 4. The primary focus of this work is the obstacle avoidance decision-making and trajectory planning phases. Therefore, assumptions are made for the phases of obstacle detection, low-level control, state estimation, and waypoint generation. It should be noted that all of these components impact the system performance during testing and training, however the purpose of this paper is to provide a baseline for quickly implementing an obstacle avoidance module for a low-cost system, given that these other components are pre-selected.

C. Imitation Learning Framework

In recent years, autonomous systems have increasingly leveraged reinforcement learning for planning and control. However, there is an inherent risk when RL is applied to aerial systems, or in obstacle avoidance scenarios. For one, real-world training can be a stressful experience that may need additional monitoring to prevent crashing. Simulation-based training can alleviate the danger of physically damaging the system, however this still requires steps of fine-tuning initial parameters, or using some guided-policy to start acquiring informative state-action pairs. As a way to circumvent these implementation complications, imitation learning trusts an expert policy generator to provide demonstrations to learn a policy.

Expert trajectories provide a means of training a learned policy for a specific task, such as trajectory planning. However, it is now uncertain as to whether this trajectory generalizes to scenarios, which the expert has not provided data. If all scenarios were known and could be solved by an expert trajectory off-line or on-line, then there would be no need for this learned policy. One example of addressing this was the work by Pfeiffer et al [10], where model-free reinforcement learning was used after training from an expert. However, this still requires letting the system, or agent, operate in the environment with an unknown policy network. A solution to this dilemma is presented in the DAgger framework, presented in [16]. The DAgger framework parametrized a policy by θ to learn from a trajectory of expert data, while allowing for online learning. The key technique is that the control system will have an increasing probability

Process	Rate (avg.)
Point Cloud + A*	5 Hz
MPC Rates	50 Hz
PID Motor Commands	100 Hz

Table 1 Obstacle Avoidance Expert Processing Speeds

of querying the learned policy at each epoch, defined by β, and the datasets are aggregated at each iteration. Therefore, the policy is learning from expert action-state pairs using onboard imagery, while also expanding the distribution of state-action pairs from applying the learned policy randomly. The probability approaches zero as the epochs approach infinity, meaning the learned policy will be increasingly used instead of the expert. This method is used to have the benefits of safe training and a wide distribution of state-action pairs. With each new iteration of training, additional state-action pairs are seen from the changes in the environment and the exploration from the policy actions.

V. Numerical Simulation

A. Experiment Setup

A series of experiments are setup for producing training data and testing the policy network. A quadrotor is initialized in an environment with a set of obstacles. The quadrotor is directed towards a target position which lies in the plane between the obstacle and itself. The RotorS[¶] simulation framework developed with the Gazebo simulation environment is used for constructing the quadrotor models and the controller architectures. They are installed on a PC with Ubuntu 16.04 and ROS-Kinetic[‖].

One of the major limitations of obstacle avoidance algorithms is processing power and speeds. In Table 1, the rates of each of the processes for the expert are shown. The point cloud processing and A* solution have the longest average processing time. It should be noted that the A* algorithm used in this work was not optimized for speed nor efficiency. However, the problem for the A* algorithm is rather simple and a large increase in processing time was noticed from slight changes in the environment. For this reason, the DNN was designed to map the raw images to the 2D waypoint in place of the A* algorithm. This also allows the point cloud formation and handling process to be removed when applying the learned policy, since it only needs the raw image information.

The Deep Neural Network (DNN) is setup using PyTorch[**] with a Convolutional Neural Network (CNN) and additional fully-connected layers. The CNN features four convolution layers and three fully-connected layers with the ReLU activation function. A range of CNN parameters were tuned to find the most efficient model for learning the avoidance maneuvers, while being computationally feasible. Both grayscale and RGB images were tested. In addition, relu and elu activation functions were compared. Lastly, the size of the input image varies between 32x32, to 256x256. The least computationally expensive, yet successful network was the one shown in Figure 5, with a 64x64 RGB image and Relu activation function. The relative target position in the 2D plane is concatenated with the features from the CNN before the additional fully-connected layers. The output is two values, the location in x and y, as seen in Figure 5. The goal was to keep a network which could be trained and retrained quickly, therefore requiring a small network with few parameters. The training process used the adam optimizer, a form of adaptive stochastic gradient descent, and mean-squared error for the loss function.

The data collection process involves initializing the quadcopter in a random location with a random assortment of obstacles between it and the target waypoint. The image collection is processed at 30 frames-per-second, and whenever an image is collected all other information is recorded, including the most recent A* path solution. The quadcopter follows the trajectory, with varying actions from the learned policy and the expert, and once it is complete the process is restarted. This makes the data collection process simple and fast. A data pipeline is used to clean the data to get rid of any errors and organize for the deep learning framework. Example data can be seen in Figure 6. The Model Predictive Control design was made particularly to produce aggressive reactions for agile flight around obstructions. The learned policy is trained with ground truth state feedback, but tested with an odometry model using inertial and GPS feedback.

A set of scenarios features obstacles in front of a quadcopter as it attempts to move forward 5 meters. Initial

[¶] https://github.com/ethz-asl/rotors_simulator

[‖] http://wiki.ros.org/kinetic

[**] https://pytorch.org/

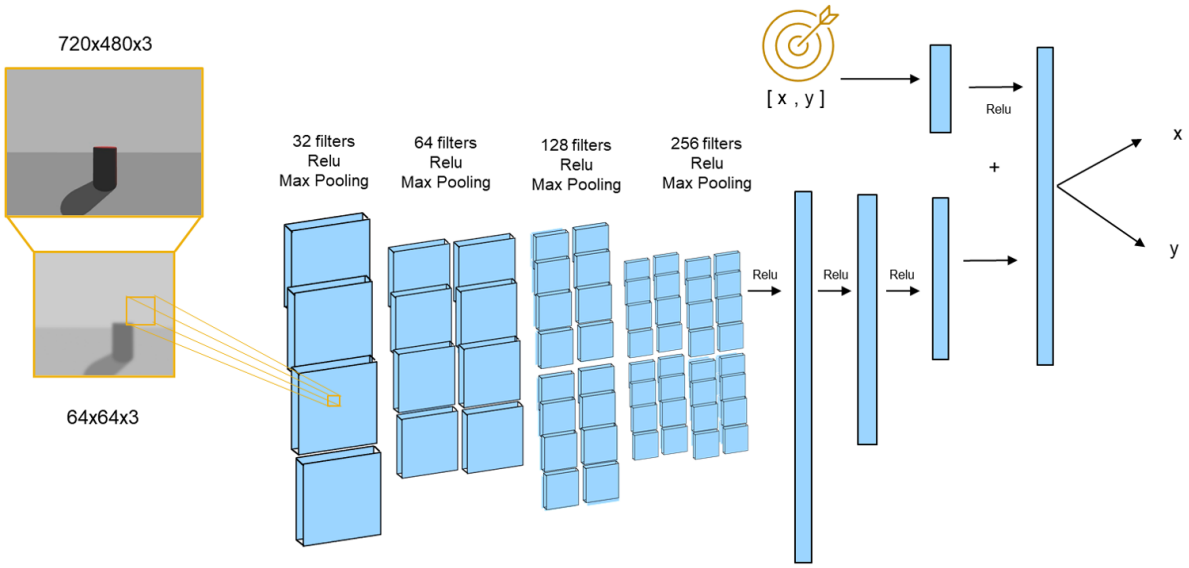

Fig. 5 Deep Neural Network Architecture

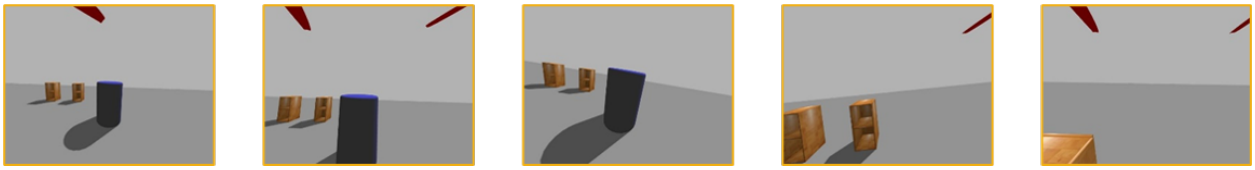

Fig. 6 Images during DAgger Training Runs

experiments used two scenarios to collect expert training data. Four iterations of training were done, with additional data being aggregated at each iteration for retraining. The DAgger method was used to increasingly use the policy during training in order to expand the distribution of states and actions of the training dataset. The final dataset featured around 30,000 images and corresponding states and actions. The images used were 64x64 RGB, and the hyper parameters used were a batch size of 64 and a learning rate of 0.005. The NVidia Quadro M1200 GPU with 4 GB of RAM was used for training, which resulted in converged results after about 30 epochs and 2 hours for the final training iteration. All the training results can be seen in Table 2. This means that high-performance clusters and millions of training data points are not needed for the successful training of this model. It should be said that there is still a complex tradeoff in model complexity and capability when generalizing to a larger subset of scenarios.

B. Simulation Results

A set of scenarios was setup to obtain expert trajectory data for training and then to evaluate the DNN policy in testing. Two scenarios can be seen in Figure 7. The first scenario includes two blue cylinders at positions [1.5, -0.5] and [4.0, 1.5]. The second scenario includes one blue cylinder at [1.5, 0.0] and two wood-textured cabinets at [3, -2.0] and

Table 2 DAgger Training Results

Iteration	Dataset Size	Scenario	DAgger	Testing Loss	Training Time	Epochs
1	4289	1	$\beta = 0.00$	1.60e-2	14 min	10
2	9830	1	$\beta = 0.25$	1.34e-2	18 min	10
3	19441	1,2	$\beta = 0.30$	2.92e-3	1 hr 15 min	30
4	33420	1,2	$\beta = 0.40$	3.92e-2	2 hr 30 min	30

[3,-3.5]. These two scenarios allows the policy to learn from a range of obstacle locations in the image plane. The positions used by the policy are relative to the goal location, which in these cases is [5.0, 0.0]. Therefore, if the goal location is changed, the policy should be able to continue to travel through the clustered environment.

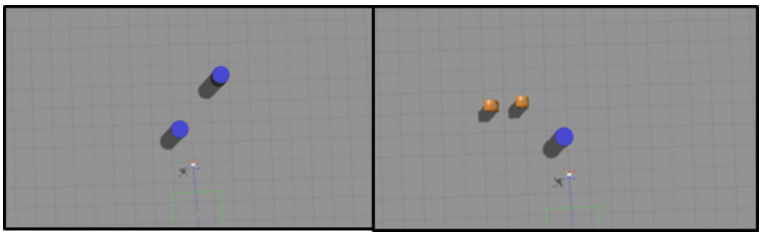

Fig. 7 Two scenarios for training and testing the policy; Scenario one includes two blue cylinders (left), and Scenario two includes one blue cylinder and two wood-textured cabinets (right)

The policy and expert were compared over the set of scenarios to determine their speed and performance. The DNN appeared to act as a smoothing function to the A* solution, as the UAS consistently completed smoother trajectories around the obstacles. This featured slightly longer distances traveled and slower travel times. The additional distance actually provided a safer distance from the obstacles since the grid was not coarse as with the A* planner. The DNN used the state estimation provided by the onboard Kalman Filter, which includes noise from the IMU, whereas training was completed with ground-truth data. Therefore, the DNN is also robust to a level of noise that comes from sensors. In addition, the processing speeds of the policy and DNN were compared. The A* policy processed at around 2.0 to 9.5 Hz. This is without converting the point cloud data into the occupancy grid, which requires additional processing. In comparison, the DNN processes in 18 to 21 Hz using the NVidia Quadro M1200 GPU, a 2x to 8x improvement from the expert. The distribution of x and y positions for both the policy and expert can be seen in Figure 8, which shows the smoother trajectory from the DNN policy.

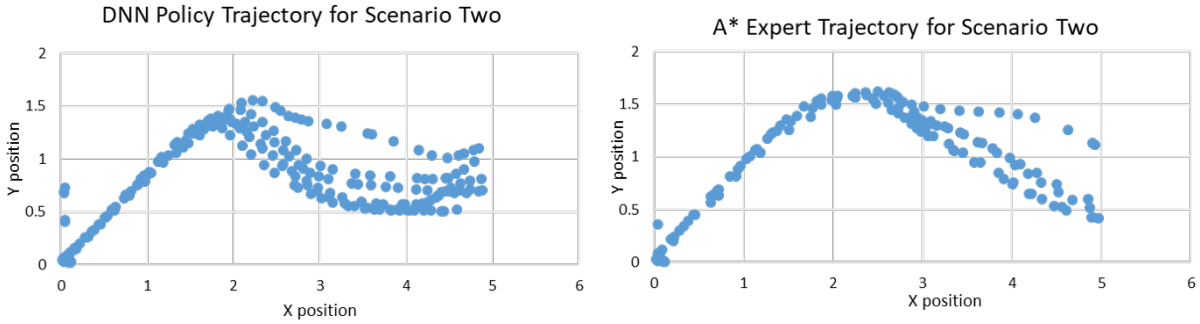

Fig. 8 Distribution of x and y position for policy and expert in Scenario Two; DNN trajectory is a smoother path (Left) as compared to the A* expert path (Right)

The implementation of this policy only requires a single monocular camera with low resolution. Therefore, the cost and weight of the sensors required is much lower than the expert, which requires a stereo camera and the additional processing units. Since a separate controller is used, and a nonlinear optimal control technique is implemented, smooth and efficient trajectories are seen during flight. Also, since the trajectory data contains the end position of hovering at the target goal the learned policy is capable of stopping, rather than having to move continuously forward. Furthermore, the policy is trained to be able to go around an obstacle rather than simply moving to the side to avoid. The previous two facts differentiate this work from earlier work.

C. Generalized Performance

The two scenarios defined before were used as standard avoidance maneuvers around a small set of obstacles. It is expected that applying the DAgger method on a small set of training scenarios can produce a trained policy that generalizes to a wide range of other scenarios. The success of theses scenarios is a combination of the successful

maneuvering around the obstacles and the speed in which the navigating is done. Therefore, a handful of random scenarios were created with a variation in the number and location of obstacles. First, a high-speed flight test exemplified how the DNN is able to rapidly make avoidance maneuvers since it only requires processing of a low-resolution image, as compares to a processing a 3D point cloud and planning in a 2D grid environment. The DNN was able to avoid in all the test scenarios, which featured 1 - 2 obstacles and where the vehicle reached a speed of 3 - 5 meters per second. The expert, however, was only able to avoid half of the cases, likely because the point cloud processing required additional time before updating the waypoint. Next, the DNN was inspected on the full set of test scenarios. Over 80% of the scenarios resulted in collision-free trajectories. This confirms the hypothesis that the trained policy would be able to generalize to a subset of unseen scenarios. Additional work needs to be done to understand the full limitations of the system.

VI. Conclusions

Obstacle avoidance is critical for autonomous systems during search missions. This is because speed is critical, and the systems either are low-cost or are utilizing high-cost sensors for other purposes. Therefore, it is important to have a method of obstacle avoidance that is quick, vehicle-agnostic, and low-cost. The method shown here features a deep neural network, trained via imitation learning to give the best safe waypoint. This is used in a framework with a nonlinear model predictive controller to create smooth trajectories. Results show great potential in training Deep Neural Networks to provide quick obstacle-free paths via raw images and relative position to a target location. Trajectories provided by the learned policy network were consistent and processed more quickly than the expert policy. This is because the expert policy requires additional processing from the point cloud to occupancy grid. In addition, the system can be applied to low-cost systems since it only requires a single low-resolution monocular camera and low-grade GPU.

The problem at hand does not have any one framework or model which will work, however a series of proven techniques and policy networks may prove useful as baselines or building blocks for more advanced autonomous systems for navigation. For that matter, there is much more work to be done to tune and improve the capability of the framework in this paper. The next step will be to make advancements on the components in the framework, then implement onboard hardware once there is enough confidence in the system's capabilities in realistic simulation environments.

The deep neural network model was specifically designed to be easy to implement and relatively cheap to train. Therefore, the framework could be massively improved by using more advanced neural network models. Furthermore, it is likely that using pretrained layers from ImageNet would promote faster learning in the first iterations of DAgger. In addition, Recurrent Neural Networks have shown success when working with sequential data. Another advancement will be to modify the 2D output to a list or waypoints in 3D to provide the MPC controller with a higher resolution and more accurate trajectory to follow. This may lead to substituting the A* expert planner with another more advanced algorithm, however the onboard computational requirements during training must be considered. In addition, the next steps will include collecting a larger dataset of trajectories with further obstacle types, background textures, and target locations, while continuing the DAgger training process over these scenarios.

References

[1] Kumar, V., and Michael, N., "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, Vol. 31, No. 11, 2012, pp. 1279–1291.

[2] United Nations Office for the Coordination of Humanitarian Efforts, "Unmanned Aerial Vehicles in Humanitarian Response," 2014.

[3] Choi, Y., "A Framework for Modeling and Simulation of Control, Navigation, and Surveillance for Unmanned Aircraft Separation Assurance," Ph.D. thesis, Georgia Institute of Technology, Atlanta GA USA, 8 2016.

[4] Choi, Y., Jimenez, H., and Mavris, D., "Two-layer obstacle collision avoidance with machine learning for more energy-efficient unmanned aircraft trajectories," *Robotics and Autonomous Systems*, Vol. 98, 2017.

[5] Barry, A., Florance, P., and Tedrake, R., "High-speed autonomous obstacle avoidance with pushbroom stereo," Journal of Field Robotics, 2017.

[6] Hrabar, S., "3D Path Planning and Stereo-based Obstacle Avoidance for Rotorcraft UAVs," 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.

[7] Michels, J., Saxena, A., and Ng, A., "High Speed Obstacle Avoidance using Monocular Vision and Reinforcement Learning," 22nd International Conference on Machine Learning, 2005.

[8] Abbeel, P., Coates, A., Quigley, M., and Ng, A., "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," Neural Information Processing Systems (NIPS) 19, 2006.

[9] Pan, Y., Cheng, C., Saigol, K., Lee, K., Yan, X., Theodorou, E., and Boots, B., "Agile Autonomous Driving using End-to-End Deep Imitation Learning," Conference of Robotics, Science, and Systems, 2018.

[10] Pfeiffer, M., Shukla, S., Turchetta, M., Cadena, C., Krause, A., Siegwart, R., and Nieto, J., "Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations," IEEE Robotics and Automation Letters Volume 3, 2018.

[11] Zhang, T., Kahn, G., Levine, S., and Abbeel, P., "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search," 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016.

[12] Kaufmann, E., Loquercio, A., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D., "Deep Drone Racing: Learning Agile Flight in Dynamic Environments," 2nd Conference on Robot Learning (CoRL), 2018.

[13] Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., and Hebert, M., "Learning Monocular Reactive UAV Control in Cluttered Natural Environments," *CoRR*, Vol. abs/1211.1690, 2012.

[14] Giusti, A., Guzzi, J., Ciresan, D., He, F.-L., Rodriguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., Scaramuzza, D., and Gambardella, L., "A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots," *IEEE Robotics and Automation Letters*, 2016.

[15] Kamel, M., Burri, M., and Siegwart, R., "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," 20th World Congress of the International Federation of Automatic Control, 2017.

[16] S. Ross, G. G., and Bagnell, A., "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," 14th International Conference on Artiticial Intelligence and Statistics, 2011.